Kid Krypto

An Introduction to Cryptography

Final Report

May 4, 2007

Group SD0617
Deepak Agarwal
Ben Anderson

Advisor
Dr. Raj Katti

Table of Contents

Background

Math is a difficult topic for educators to convey in a way that is intriguing, stimulating, and gripping. Many times the topic is taught in a 'quick dose' style that demands quick answers to quick problems and does not foster extended concentration on a problem. Reasoning is replaced by the faster rote style of learning, and the deep and interweaving world of mathematics is lost in a shallow list of rules and legalism.

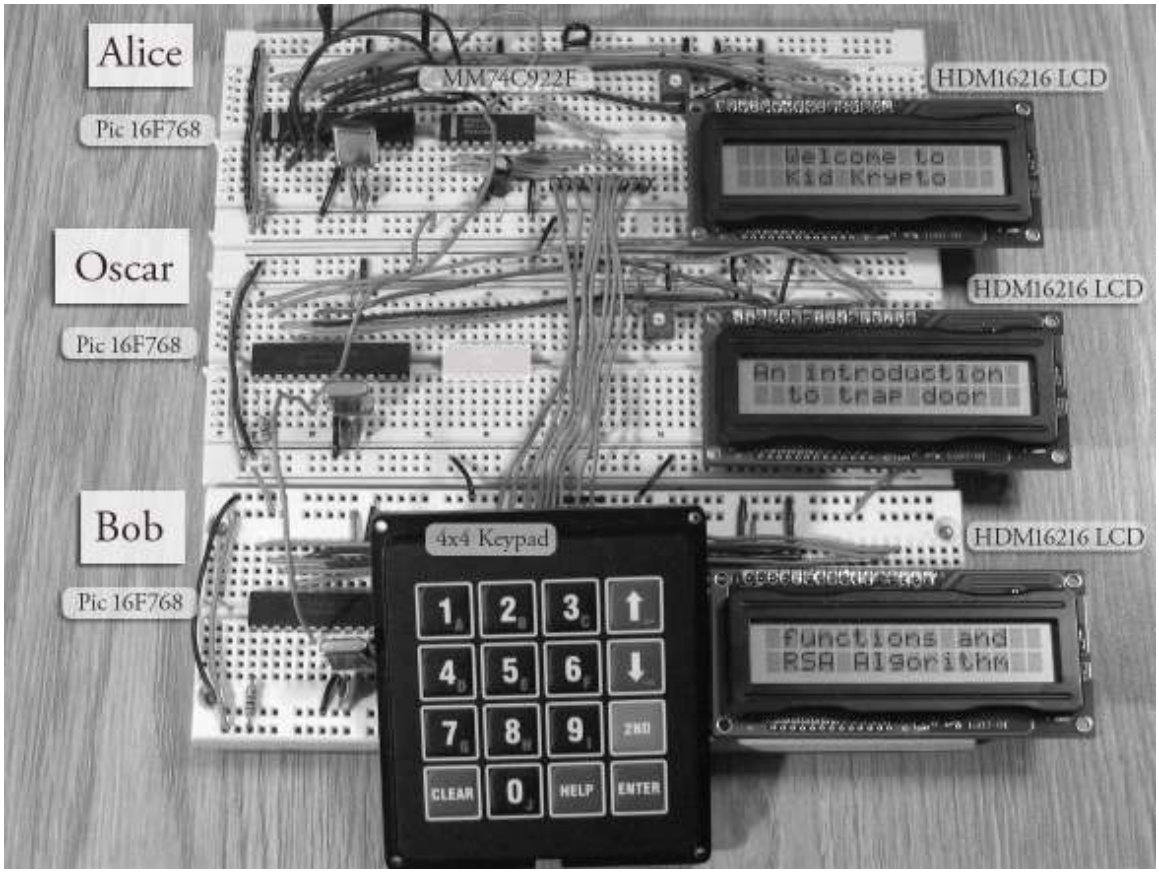
An effort to bring math education to a new level has been made by Michael Fellows and Neal Koblitz in their paper, "Combinatorially Based Cryptography for Children (and Adults.)" They believe the use of cryptography in early education can be of great benefit to students, showing them that math does not have to be a dry repetition of times tables, but a challenge to meet, a hurdle to mount, and even an adversary to overcome. The use of ciphers and codes and how they are made can breathe life into an otherwise dusty subject.

The project we propose to develop will be a learning tool that will convey a foundational concept of cryptography (and ultimately, mathematics) that any visiting middle school, high school, or college student could use while at NDSU and come away with a new knowledge of cryptography and perhaps a fresher view of mathematics. Our intention is for it to be a standalone display (like something one might find at a science museum) that will present its concept in an entertaining and rewarding way.

Requirements

1. We will produce a device and presentation that will convey a fundamental cryptographic concept (and its mathematical grounding) to students from grades 7 – 12 and beyond in a challenging, thought provoking, and entertaining way.
2. We will gain much knowledge on the topic of cryptography and its educating potential for math students in elementary school through high school.
3. The key features of this project will be
    a. A circuit board based electronic device with 3 LCD display and interactive controls.
    b. A display background will be necessary to provide instruction on using the device, the process, theory, and applications. It will provide a background of cryptography and its use in society today.
    c. An interactive program that teaches the fundamentals of cryptography via the RSA algorithm with graduated examples of increasing difficulty.
4. The project will be focused towards NDSU tour groups, so time will be an issue. We want the total program time to be no more than 20 minutes. Also, it must have a minimal or zero learning curve to operate.
5. As this will be used for public display, it will be a reflection on the quality of NDSU's engineering department. Therefore, the project itself must be of quality build, design, and presentation. Nice to look at and easy to interact with. We want it to last as long as we can make it last.

Hardware



Our layout consists of three boards: Alice, Oscar, and Bob. They represent the Sender, Eavesdropper, and Receiver, respectively. Each has a corresponding PIC and LCD. Alice also includes a keypad and decoder to interpret user input. Alice controls the flow of the program and Oscar and Bob perform support roles in the operation.

Please see the Appendix for data sheet references to parts used.
Please see the Appendix for wiring schematics.

Design Implementation Options

Though we've had a fairly straightforward impression of what is needed in this project from early on, our first thoughts on how to do this project considered several different options:

Strictly Hardware

At the very foundation of our project is the concept that cryptography for education is a 'paper and pencil' type of project. A young person can learn valuable concepts from puzzles or problems solved simply with a pencil and paper. An early idea we had was to somehow make an interactive hardware device that made use of an array of pressure sensitive LED's to be an example of such a paper and pencil problem. An advantage to doing the project in this way is that it holds closely to the fundamental document that inspired this project. To use a simple, hands-on project in a way that is fairly elementary to understand (yet rather sophisticated to implement) is the heart and soul of the project. The disadvantage to such a design is user friendliness. There would be very little means of communicating the point of the device to a passerby except by means of a poster board display to explain it. At that point, the reading becomes a bit of a hang-up for someone who is not so much interested in spending 20 minutes studying a poster trying to figure out how the device is supposed to work. We would prefer that the device itself is more interactive and self-explanatory.

Strictly Software

Another option we considered at the onset was to go to a completely software platform. We would write a program that could be installed and run on a standard computer terminal that would pass along our concepts on cryptography. Some advantages to this idea included the inexpensiveness considering it would be wholly software, the complete walk-through nature a program can have, the relative ease of implementing it as a project compared to designing an electronic device (the three of us are computer engineers and perhaps a little more comfortable with the programming,) and also the potential portability of the project. (i.e. The program could be stored on a floppy or CD and installed by a tour leader at a computer, copies could be distributed as souvenirs, etc.) The disadvantages we believe to exist lay in the idea that if this project were to be used as a tour point in the future, it might be lacking the inspiration to hold a tour group's attention. Educational software can be effective, but it lacks the punch that an actual device you can see and work with has. In all other respects, this implementation would be fairly ideal. An idea we are considering (depending on the success of our main

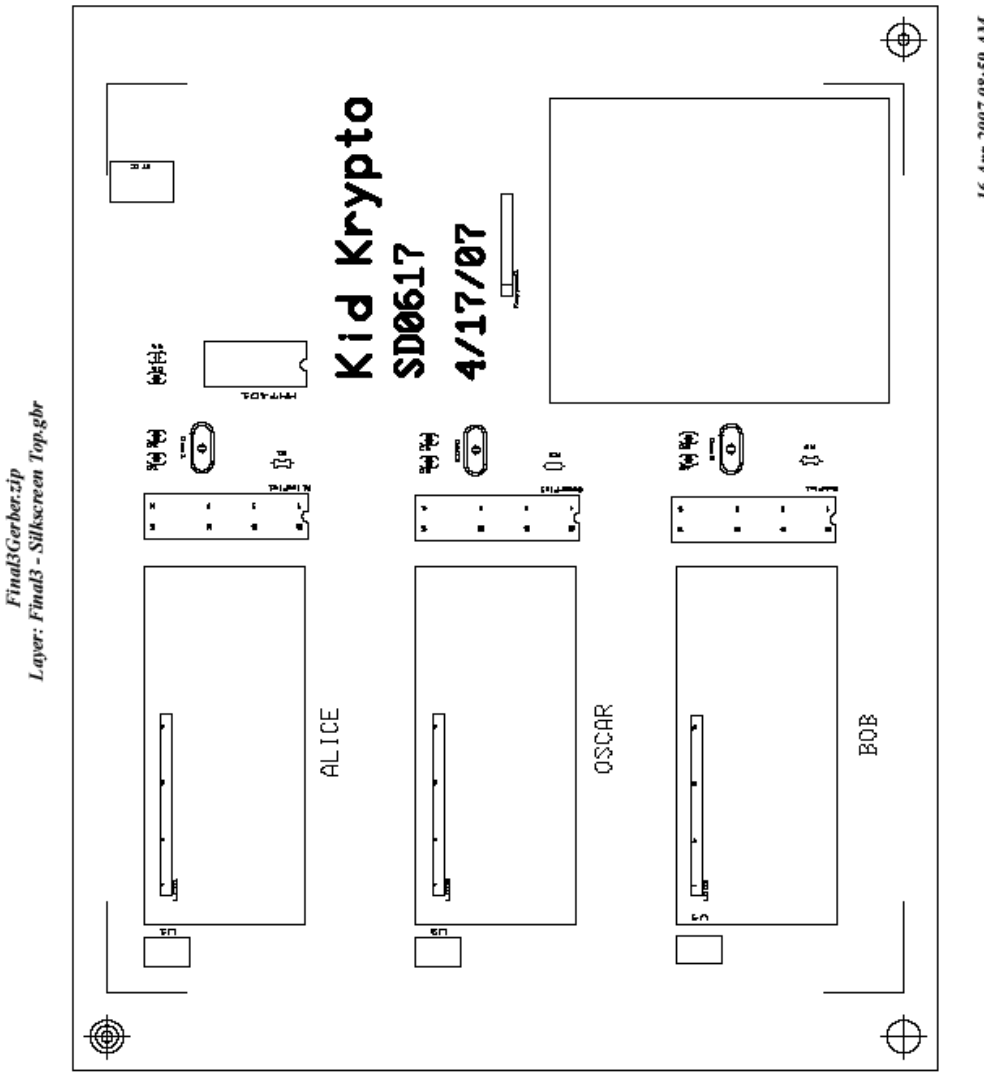
implementation) is to develop a software tutorial as a supplement to the device and the two can be used hand in hand.

PIC Processor Evaluation Board Implementation

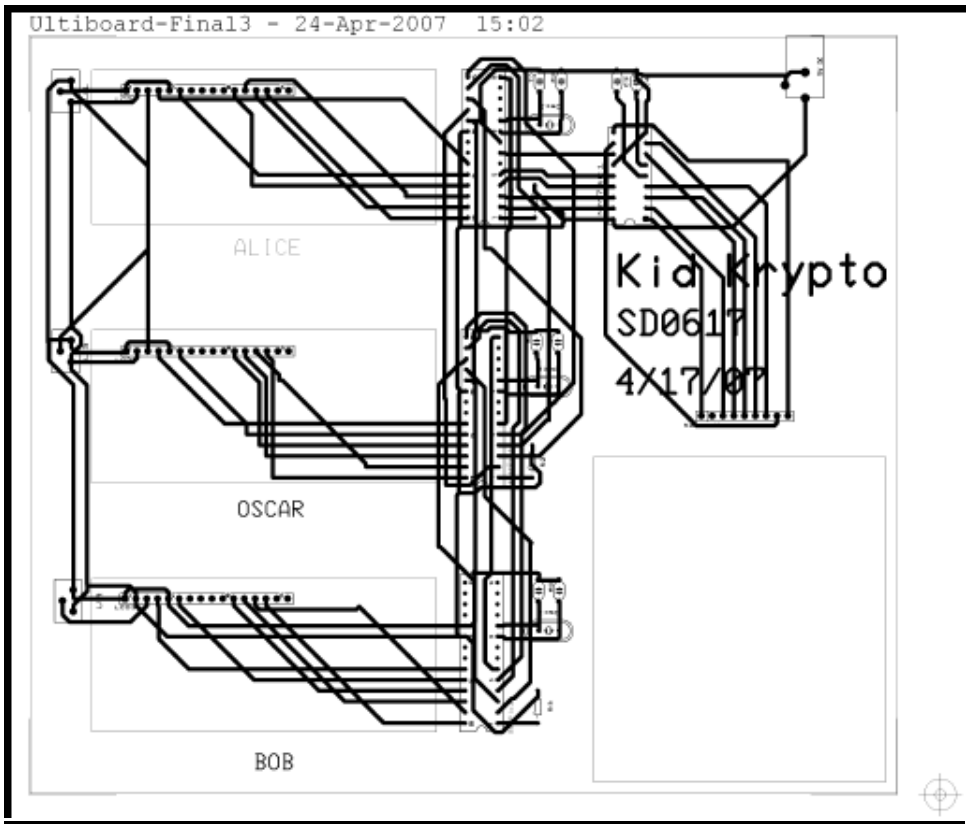
The option we consider most viable is the mix of software and hardware we can find in the NDSU PIC Processor Board unit used in ECE 376 Embedded Systems. The advantages to such a device include the 'romanticism' (if you can call it that) of actually having a circuit board loaded down with open wires and capacitors and seeing the 'guts' of the project. Students would be working directly with this object that has been actually put together by their predecessors. Using the same type of processor evaluation board that they could be using in the future also adds to the interest. Along those same lines, as this is the board used for class, it is readily available and we know it works. Plus, it makes a good base to start from as our group is already acquainted with it to a small degree. The unit would still require a heavy dose of programming which plays to our strengths as computer engineers. It would be portable and as self explanatory and user friendly as we are able to program it to be. Disadvantages exist for this design as well. There will only be one of these units and if it breaks, we are very set back. The potential for multiplicity and distribution that the software has goes down greatly. Only one person can directly use the device at a time. In the end, however, we believe this will be our most effective route in conveying cryptographic concepts to tour groups.

PCB Layout

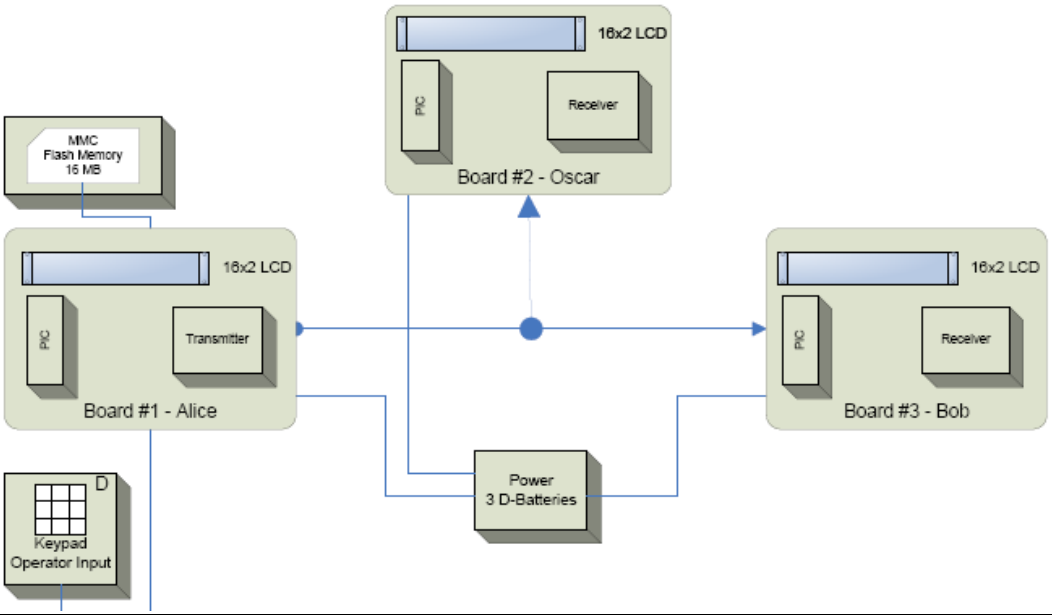
Fully implemented, the system would be mounted on this PCB layout. This is the Top Silkscreen layer approved by www.4PCB.com.



This is an Ultiboard representation of the top and bottom copper layers of the PCB layout.



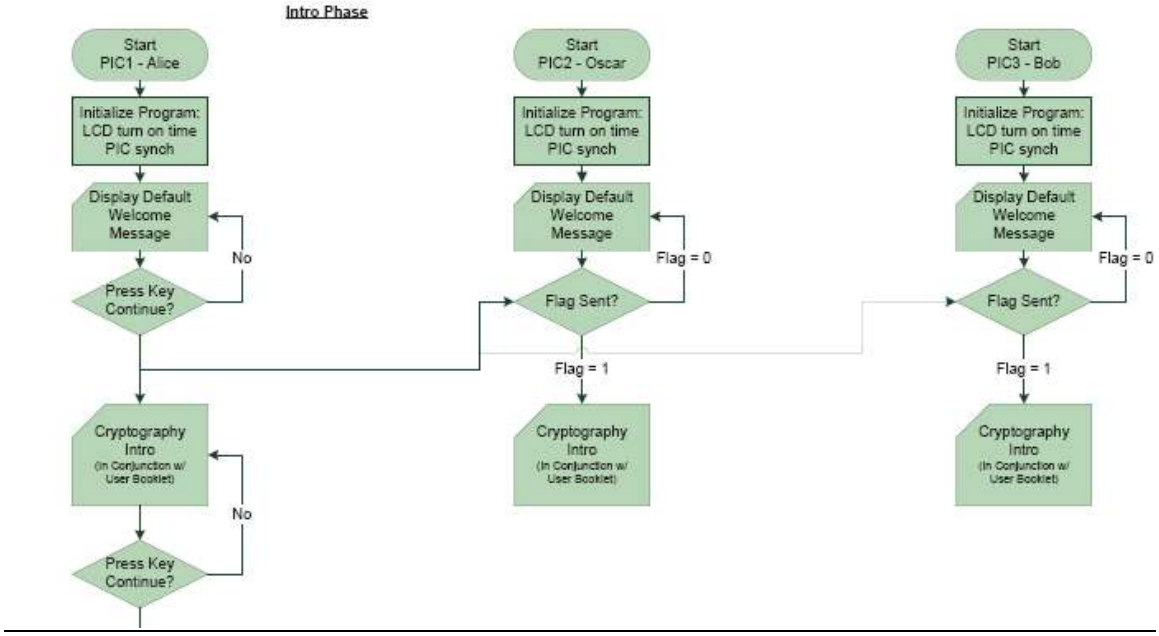
An original block diagram of how the Kid Krypto system would be laid out.

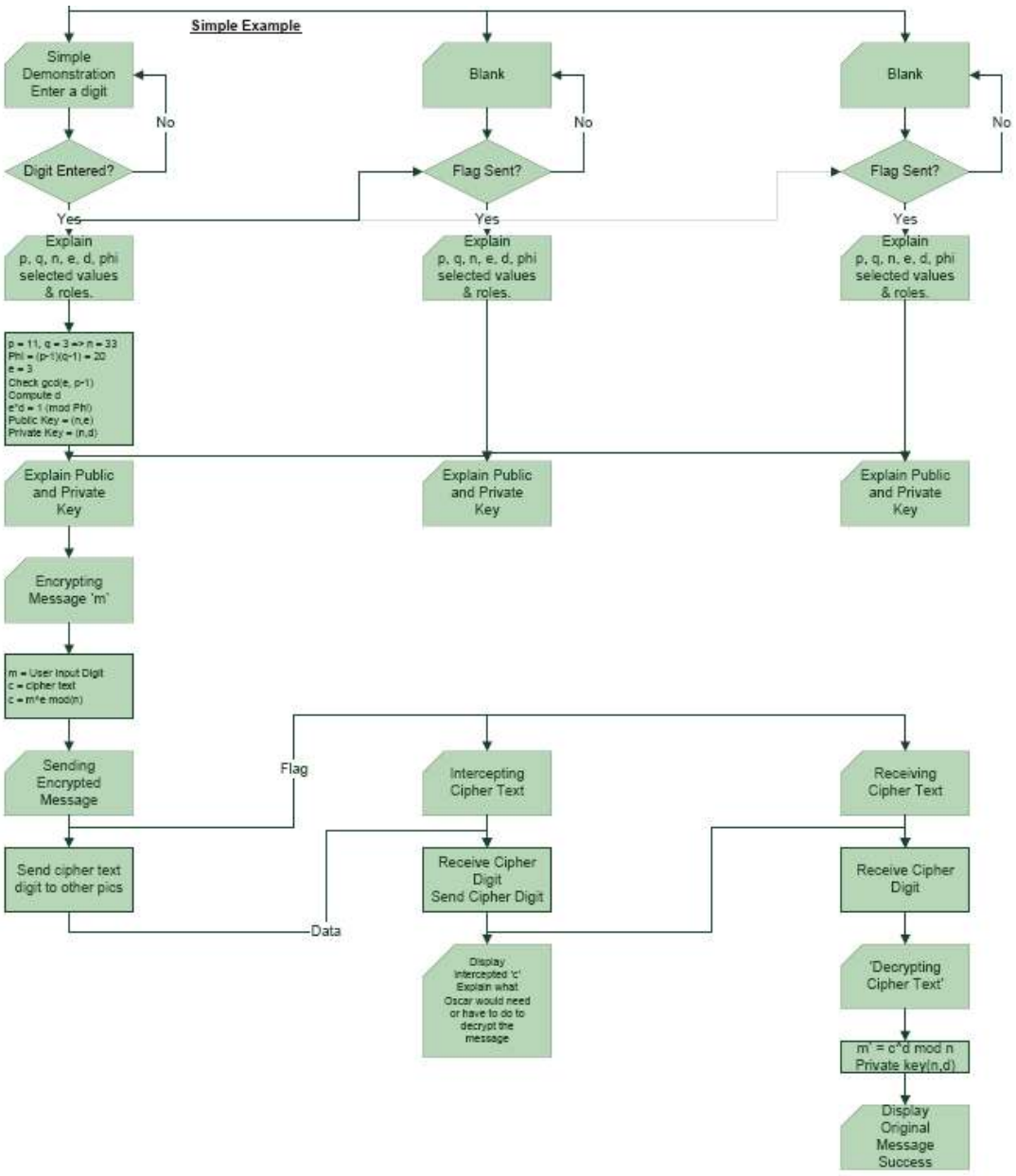


Software

Flow Chart

A flow chart overview of how the software runs and interacts with the user and itself.




Simple Example
Simple Demonstration Enter a digit
No
Digit Entered?
Yes
Explain p, q, n, e, d, phi selected values & roles.
p = 11, q = 3 => n = 33
Phi = (p-1)(q-1) = 20
e = 3
Check gcd(e, p-1)
Compute d
e*d = 1 (mod Phi)
Public Key = (n,e)
Private Key = (n,d)
Explain Public and Private Key
Encrypting Message 'm'
m = User Input Digit
c = cipher text
c = m^e mod(n)
Sending Encrypted Message
Send cipher text digit to other pics
Blank
No
Flag Sent?
Yes
Explain p, q, n, e, d, phi selected values & roles.
Explain Public and Private Key
Flag
Intercepting Cipher Text
Receive Cipher Digit
Send Cipher Digit
Display Intercepted 'c' Explain what Oscar would need or have to do to decrypt the message
Data
Blank
No
Flag Sent?
Yes
Explain p, q, n, e, d, phi selected values & roles.
Explain Public and Private Key
Receiving Cipher Text
Receive Cipher Digit
'Decrypting Cipher Text'
m' = c^d mod n
Private key(n,d)
Display Original Message Success

Algorithms

The Miller-Rabin Algorithm

The Miller-Rabin algorithm uses **Fermat's Theorem:**

If p is prime and a is a positive integer not divisible by p, then:

$$a^{p-1} = 1 \bmod p$$

or another useful form:

$$a^{p} = a \bmod p$$

The mod function, short for the modulo function, is also known as the remainder function (or residue). In other words,

a mod p is defined as the remainder when a is divided by p.

For example:

$a = 7 \qquad p = 5$

$a \bmod p = 7/5 \qquad \Rightarrow 1$ remainder 2

$a \bmod p = 2$

An interesting side venture is determining the formula for the mod function, but we'll leave that discovery up to the reader.

Surprisingly, the Millier-Rabin algorithm does not necessarily yield a prime number, but like almost all other primality tests, the yielded number is *almost* certainly a prime number due to the fact that this algorithm works off of probabilities.

How it works is like follows:

1.) Find integers k, q with $k > 0$, q odd, so that ($n-1 = 2^k q$)
2.) Select a random integer a, $1 < a < n - 1$
3.) If a^q mod $n = 1$ then our test is inconclusive (possibly prime)
4.) For all a's 0 to k-1 check that a^{2jq} mod $n = n$-1. (still possibly prime)
5.) If at any time a^{2jq} mod n is not equal to 1 or n-1, then it is not prime.

For Example:

If we were to test if 15 were prime, we would find

$n = 15$ $n - 1 = 14$ $14 = 2^k q \Rightarrow (2^1)*7$

$k = 1$ $q = 7$

We pick $a = 4$. Then we have

$4^7 \bmod 15 = 4$ => NOT Prime

The 4 we found is neither 1 nor 14, which shows that 15 is not prime.

However, if we would have taken $a = 14$, our equation would be

$14^7 \bmod 15 = 14$ => POSSIBLY Prime

The 14 returned indicates that 15 is possibly prime. For all other values of *a* (0-13) we find that 15 is not prime, and it only takes one test of 15 not being prime to show that it is not prime. So, if we were to only test 15 once, we would have a 7.1% chance of returning 15 as a prime. The more tests there are with different values of *a*, the more drastically that percentage is reduced. In the case of $n = 15$, two tests using two different values of *a* would leave no chance of error. For larger values *n* that aren't prime, there are more chances that *n* could be shown as prime. For instance 13 * 17 = 221, and there are six values of *a* that would show 221 is possibly prime. At some point, there is no way you can check every value of *a* and have to set how many tests you would like to make. A sufficient number of tests that do not return *n* as a possible prime number will *almost* certainly show that *n* is a prime number.

In practice, 75+ digits *p* & *q* are randomly chosen and tested for primality. This can take a long time, but a public and private key only need to be generated once every so often, so the time factor is acceptable. Once they are found, they are multiplied to find *n*.

Calculate the Euler Totient

The RSA Algorithm uses a relationship that is described by a corollary of Euler's theorem:

Given two prime numbers *p* and *q*
Given two integers $n = pq$ and $m < n$
Given an arbitrary value *k*

$m^{k(p-1)(q-1)+1} = m \bmod n$

Phi (or Φ in Greek notation) is used to represent $(p - 1)(q - 1)$, which is also known as the Euler Totient.

$$\Phi(n) = (p - 1)(q - 1)$$

The Euler Totient represents the number of positive integers less than n and that n cannot be divided evenly by. (How many numbers leave remainders when n is divided by them.) We can break Euler's equation down to the following:

$$d = e^{-1} \bmod \Phi(n)$$

Where e and d are multiplicative inverses of each other AND relatively prime to $\Phi(n)$. (Their greatest common divisor is 1.)

Choose *e*

Euler's Corollary: $d = e\text{-}1 \bmod \Phi(n)$

Generally, e is a standard large prime number that is relatively prime to $\Phi(n)$ and less than $\Phi(n)$. In our case, we want to choose e low to keep number sizes manageable. Remember your number phi? We need it to choose an appropriate value e. We need e to be relatively prime to phi, so choose e so that phi does NOT divide evenly by it.

Calculate the Multiplicative Inverse

We can find the multiplicative inverse by using this extended Euclid function.

EXTENDED EUCLID(m,b)
1. (A1,A2,A3) <- (1,0,m); (B1,B2,B3) <- (0,1,b)
2. **if** B3 = 0 **return** A3 = gcd(m,b); no inverse
3. **if** B3 = 1 **return** B3 = gcd(m,b); B2 = $b^{-1} \bmod m$
4. Q = floor[A3/B3]
5. (T1,T2,T3) <- (A1–QB1,A2-QB2,A3-QB3)
6. (A1,A2,A3) <- (B1,B2,B3)
7. (B1,B2,B3) <- (T1,T2,T3)
8. **goto** 2

Pubic Key Encryption

RSA Algorithm: Cryptotext = Message$^e \bmod n$

For a Message less than n, we can encrypt the message using the RSA Algorithm.

It is a simple matter of using the equation with our public key. Since we know our message, *e*, and *n*, we just fill in the blanks to obtain the cryptotext.

Decryption Using the Private Key

RSA Algorithm^{-1}: Message = Cryptotext^{d} mod n

Decryption is a simple matter of using our generated private key in a very similar way to how we created the cryptotext.

Code

Essential C Files:

Alice – Primary Program Driver – Keypad Interface – Encryption Method
Oscar – Secondary Program – Supplemental Info - Hacker Method
Bob – Secondary Program – Supplemental Info – Decryption Method
LCD – LCD Utility File – Wait – Initialization – Print - Move
Keypad – Interprets decoder and returns character

Most modifying programming will occur in Alice, Oscar, and Bob. LCD and Keypad are well established, but could be updated at a later time.

See Appendix for commented code.

Cost

Kid Krypto finished well under budget. Final cost was just over $100 of approximately $150 budgeted.

See Appendix for detailed budget breakdown.

Technician's Troubleshooting Section

Kid Krypto runs from a 5v source. As it is still in prototype mode, it is easy for wires to come undone but generally easy to replace by following the schematics.

Blank Screen on startup –

LCD is not initializing. Be sure PIC is plugged in appropriately and there are no loose wires from the PIC to the LCD.

Also, check the trim on the variable resistor located near the LCD to make sure it is set properly. (Should very low, almost grounded.)

Keypad not typing properly -

Be sure the keypad wire array is lined up with the decoder wires appropriately. The rightmost pin (9) should not be connected to any wire. Be sure it is not plugged in upside down. Keys should read the same way as the LCDs.

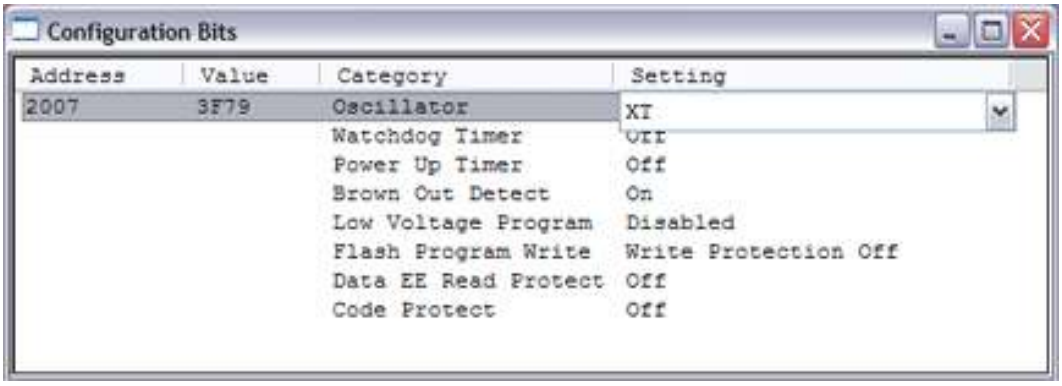
Keypad printing too fast –

Kid Krypto is programmed to run at a 4MHz clock. Be sure the crystals are 4MHz.

Reprogramming –

We use MPLab to program Kid Krypto. The compiler we used was Hi-Tech C Picc (the full version in demo mode.) This allows the full 8k of memory to be used in the PIC. We used PIC-Start Plus as our programming tool.

1. Select PIC-Start Plus as the programmer in MPLab.
2. Programmer -> Enable Device
3. Build all code after finished editing.
4. Make sure device is set for PIC16F768A (Configure -> Select Device)
5. Make sure Configuration Bits are correct



6. Insert PIC into PIC-Start Plus and program Device (Should take around a minute or two)
7. Remove PIC and reinsert into board. If all goes well, your software change will have fixed the problem.

Project Comments

Like any major projects, there were difficulties encountered in the course of production. Perhaps the biggest difficulty was the loss of one of our partners after the first semester. We were counting on his knowledge and insights to help bring this project to completion. When he left, we had to do some major reallocation of duties and planning so we could attempt to move forward in a timely manner.

The hardware we chose, while it functions to an extent, was probably underpowered for what we originally envisioned. This was shortsighted on our part. We chose components that we were already somewhat familiar with so as to ease the process of building the project and reduce frustration in learning a new technology (as we were already learning a whole new subject matter alongside building our project.)

To that end, we have run into limitations in the size of numbers we can use. In cryptography, numbers routinely exceed the maximum 32 bit variables available on a PIC. With more crafty programming and software development, this could perhaps be worked around. However, we were unable to implement that kind of code in the time allowed.

Program size is limited to 8k so we are unable to include too many features beyond the basic program. Error checking and more 'Object Oriented' functions are beyond the scope of what we can do here. However, for a straight forward program, this amount of memory does seem passable. Modules of the program also suffered from size limits. A block of machine code cannot exceed a certain size (known as a psect) before the PIC will run out of room in a memory bank. Code must be divided so that it can be compiled into machine code and placed into different memory banks.

Using HI-Tech C PiccLite was troublesome initially, due to the 2k compile limit and limited devices supported. This was a major frustration originally, but once we moved to the full version of the HI-Tech C compiler (with a 45 day demo) most of our problems were relieved and we could focus on programming.

The programming was not finished in time to implement the project onto a PCB board. Some more refinements should be made before the project is fit for public consumption. Input verification, error checking, and exception handling all should be implemented first. Also, the program should be wrapped up more nicely to show the differentiation between Alice, Oscar, and Bob and their roles in the operation. This project is highly dependent on the user guide, but it would be nice to ultimately get away from a paper document that needs to follow the tool around.

We learned a lot about project management and the lifecycle of a project. The conceptualization process, gathering information from the customer, requirements captures, design options, progress updates, and all that it entails was greatly educational and we learned very much through that process.

We learned not to dwell on decisions after they have been made. If a decision has been made, don't waste time second guessing and wondering if it was the right one. Make a move and make it work. You can always go back if there was a mistake, but you'll never get anywhere if you don't move forward in a timely manner.

Cryptography was a totally new topic for our group when we started, and in investigating how it works we learned so much that our previous education had left untouched. Cryptography is a truly fascinating topic and we enjoyed learning about it and can see why it has potential as a teaching tool to make math more interesting.

Creating an educational device greatly opened our eyes to the difficulties that exist in trying to make complicated topics accessible and easy to grasp for someone totally new to the subject. This project seemed like a fairly straightforward task until we finally grasped that it is meant for people to learn from. That opens a whole new can of worms that delves into how people learn, what is too easy, what is too hard, and what is too boring to even sit through. Education is a matter of taste, and we have a deeper respect for those who make it their life's goal to educate, because it is a HARD thing to do.

In the future, I would like to see our project developed into a fully functional teaching device that doesn't depend quite as much on an accompanying User Guide. Larger LCD's would be effective, as well as larger microcontrollers capable of more storage and variable sizes. Perhaps a memory card could be implemented to store different programs on the board and uploaded to the PICS as the user wanted to change the program. Perhaps even having swappable memory cards would be an option.

Our advice for new design students would be: "Don't be afraid of making a few small mistakes early on when they can be changed and remedied. Be more afraid of not moving forward until you feel things are perfect. When you finally do move forward, you may realize (as you very well probably will) that the meticulously thought out plan missed a few key points and now it is too late in the game to make up for them." Also, communicate with your advisor and customer often and keep them updated. Their ideas for your project might change as you all become more familiar with what the project is really about.

All in all, we learned an incredible amount about project design, cryptography, education, and ourselves in this project. These are all valuable lessons that we will be able to take with us into the future.

Appendix